UnG-MoCha: Neural Motif Counting in Uncertain Graphs

Lujie Ban
lujieban@link.cuhk.edu.cn
The Chinese University of Hong Kong, Shenzhen
School of Data Science
Shenzhen, Guangdong, China

Xiaolin Han*
xiaolinh@nwpu.edu.cn
The Northwestern Polytechnical University
School of Computer Science
Xi'an, Shaanxi, China

Jinyang Li
jl0725@connect.hku.hk
The University of Hong Kong
Department of Computer Science
Hong Kong, Hong Kong

Chenhao Ma*
machenhao@cuhk.edu.cn
The Chinese University of Hong Kong, Shenzhen
School of Data Science
Shenzhen, China

Abstract

Motif counting is fundamental in graph analytic tasks (e.g., clustering and recommendation) but #P-hard. Recent research has focused on exploring and applying deep learning-based solutions to tackle this problem. However, these solutions assume a deterministic graph where edge existence is certain, which may not hold due to the measurement and statistical prediction errors. Meanwhile, existing methods for uncertain graphs still face considerable time costs. To address the above issues, we propose **UnG-MoCha**, a novel deep-learning approach to efficiently count motifs in uncertain graphs. UnG-MoCha extracts representative subgraphs via graph structure learning and learns graph and motif representations using hierarchical and classic graph neural networks, respectively. Canonical correlation analysis is used to exploit correlations between graph and motif representations, boosting accuracy. Experiments on real-world graphs demonstrate UnG-MoCha's superior performance for scalable motif counting on uncertain graphs.

CCS Concepts

• **Computing methodologies** → **Neural networks**; • **Information systems** → **Data mining**.

Keywords

Uncertain Graph, Motif Counting, Graph Neural Network

ACM Reference Format:
Lujie Ban, Xiaolin Han, Jinyang Li, and Chenhao Ma. 2025. UnG-MoCha: Neural Motif Counting in Uncertain Graphs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3711896.3737170

*Xiaolin Han and Chenhao Ma are Corresponding Authors.

KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1454-2/2025/08
https://doi.org/10.1145/3711896.3737170

1 Introduction

Graphs are prevalent to model the complex relationships among different objects in various domains, e.g. bioinformatics and social science, and graph analytic tasks have attracted much attention. Motif counting is a fundamental problem in graph analytic tasks. Motif, or graphlet, is usually a small graph with a few nodes and edges [44]. The motif counting task is to count the number of motifs in a graph by subgraph isomorphism and provide an insight to mine the complex graph. It has found various applications in a wide range of fields, including biological network analysis [54, 55], social network analysis [16, 23, 27, 70] and graph database query optimization [38]. For instance, in graph databases, motif counts are helpful for identifying efficient query execution plans to optimize query performance [38].

Numerous algorithms have been devised to tackle the motif counting problem effectively, falling broadly into two categories: enumeration methods [6, 29] and analytical methods [4, 43, 48]. The former often struggle with large graphs due to the NP-Complete nature of checking motif existence through isomorphism [6] and the #P-hardness of counting motif appearances via isomorphism [29]. In contrast, analytical methods, which decompose a motif into smaller subgraphs and derive the motif's count based on counts from these subgraphs, often face limitations in motif sizes and generalisability. In general, exact counting solutions [4, 6, 43, 48] are suitable for small graphs, but their performance declines for larger graphs. Consequently, researchers have turned to approximation algorithms [8, 12, 49] to enhance efficiency. Learning-based counting algorithms [39, 58, 61, 67] have been proposed and attracted much attention in recent years. These methods treat the counting task as a regression problem, delivering impressive accuracy and faster query processing compared to traditional approximate methods.

Uncertain Graphs. Predominantly, existing motif counting methods concentrate on deterministic graphs, omitting consideration for the uncertainty of edge existence. However, modeling the data as deterministic graphs is not appropriate in some applications [19, 31, 34, 50]. In protein-protein interaction (PPI) networks, the edge indicates the interaction between two proteins and the node denotes the protein compound. Here, the edge existence is uncertain because the evidence of protein interaction can be erroneous based on experimental results [5]. In user-item networks, the edge existence probability can indicate the likelihood of a user

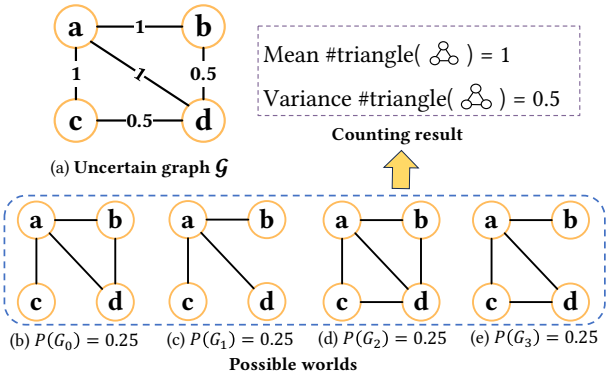


Figure 1: Uncertain graph counting example.

making a purchase of an item. In PPI networks, motif counts can be employed to analyze clustering coefficients and unveil functional and topological characteristics, as highlighted in [60]. Similarly, within uncertain user-item networks, motif counts, particularly 4-cycle counts, could be utilized to recommend items that enhance the cohesion of strong communities, as discussed in [69].

As shown in Fig. 1(a), in uncertain graphs, edge existence, influenced by existential probability, necessitates a different analytical approach to avoid imprecise analyses for motif counting. If we ignore the uncertainty, the triangle count would be 2, i.e., $\{a, b, d\}$ and $\{a, c, d\}$ are two triangles. However, with edge existential probability considered, the uncertain graph can have four possible worlds [3] according to the edge existential statuses. The lower part of Fig. 1 shows the possible worlds and their respective probabilities. Based on the probability and the triangle count of each possible world, we can compute the mean is 1 and the variance is 0.5, which is quite different from what the count obtained by ignoring uncertainty.

To solve the motif counting problem on uncertain graphs, several approximate counting methods are proposed [18, 41, 54, 55]. These methods explain the counting results on uncertain graphs from a statistical view, where the mean and variance values of the motif will be considered and used to analyze the uncertain graph. Among these methods, LINC [41] shows better counting ability with relatively low time complexity and better accuracy. Specifically, LINC operates by first generating a specified number of "possible worlds" from the uncertain graph. Within each of these possible worlds, LINC counts the occurrences of motifs. Subsequently, using the motif counts obtained from all the sampled worlds, LINC calculates the mean and variance to estimate motif occurrences in the original uncertain graph. Despite the fact that LINC's time complexity is lower than other counting methods that focus on uncertain graphs, it still suffers from high sampling time to count the motifs with relatively sparse structure (e.g., 2-star or 3-star), especially on large graphs.

To alleviate the limitations above, a straightforward solution is to directly adapt the existing learning-based methods for motif counting on the deterministic graph, given their demonstrated efficacy [39, 58, 61, 67]. Specifically, we can use ALSS [67], a state-of-the-art solution among them, to count the number of motifs over uncertain graphs. In ALSS, it first decomposes the given query graph into smaller substructures by l-hop breadth-first search (BFS) search, and then learns their representations using a Graph Neural Network(GNN) model. In addition, a self-attention network is applied to aggregate all the substructure representations. Finally, an active learner is utilized to enhance the training data and boost the estimation accuracy. However, ALSS has poor accuracy on uncertain graphs. ALSS primarily relies on pre-trained task-irrelevant models, such as Node2Vec [22], to analyze the target graph, which can not well capture the important local structural information of the target graph and neglects the uncertainty associated with its edges.

Our Solution. Addressing the protracted computation time of LINC and limitation of ALSS, we present the Uncertain Graph Motif Counter by Hierarchical Architecture (**UnG-MoCha**), a novel learning-based motif counting model. The source code and datasets of UnG-MoCha are released publicly [1]. UnG-MoCha consists of four steps: subgraph extraction, uncertain subgraph structure learning, representation learning, and count estimation: (1) In *subgraph extraction*, the uncertain graph is decomposed into a collection of smaller subgraphs. (2) During *uncertain subgraph structure learning (USSL)*, the extracted subgraphs are refined to obtain the representative possible world for each of them by effectively capturing edge uncertainty. (3) The *representation learning* step encompasses two parallel processes. On one hand, the obtained possible worlds are fed into our hierarchical *target graph neural network* to learn the representation of the uncertain graph, scaling the receptive field from the entire graph to the subgraph level. On the other hand, our *motif neural network* employs the GNN model to effectively capture the motif's structural information, obtaining the motif representation. (4) Lastly, the *counting unit*, leveraging canonical correlation analysis, efficiently mines the correlation between the graph and motif representations, computing high-quality estimated results.

Compared with LINC, the *subgraph extraction* and *USSL* modules enable our model to avoid mass time cost on possible world sampling as LINC. Compared with ALSS, which applies the pre-trained embedding model to obtain the target graph representation, our *target graph neural network* can learn the representation of input uncertain graphs through the obtained possible worlds and preserve the edge uncertainty, hence improving the estimation accuracy on uncertain graphs.

Contributions. We summarize our main contributions as follows:

- We present UnG-MoCha, a novel learning-based motif counting method for uncertain graphs. To our knowledge, UnG-MoCha stands out as the first DL-based method developed for motif counting in uncertain graphs.
- We formulate Uncertain Subgraph Structure Learning (USSL), a downstream-task-relevant graph refinement strategy to leverage the edge uncertainty and find the most representative possible worlds for each uncertain subgraph.
- We incorporate canonical correlation analysis to uncover the correlation between motifs and the uncertain graph, ultimately enhancing the accuracy of our approach.

[1] https://github.com/banrichard/UnG-MoCha

- Experiments conducted on five real-world datasets underscore the effectiveness and efficiency of UnG-MoCha. The experimental results show our method is capable of handling motif counting tasks over uncertain graphs with reducing up to 90% of the errors compared with ALSS and up to six orders of magnitude less time than LINC.

2 Preliminaries

In this section, we first provide a formal definition of uncertain graphs, followed by an articulation of the motif counting problem within the context of uncertain graphs.

2.1 Uncertain Graph

Definition 2.1 (Uncertain Graph [41]). *An uncertain graph, denoted as $\mathcal{G}$, is defined as a triplet (V, E, P), where:*

- *V represents the set of nodes in the uncertain graph,*
- *$E \subseteq V \times V$ constitutes the edge set,*
- *$P : E \rightarrow (0, 1]$ is a function that assigns to each edge $e \in E$ a probability $P(e)$, indicating the likelihood of the existence of edge e.*

When the edge existence probabilities are disregarded, the uncertain graph reduces to a deterministic graph $G = (V, E)$. This deterministic graph is referred to as the *backbone graph* [13] of the original uncertain graph $\mathcal{G}$.

According to the *Possible World Semantics* (PWS) framework [3], an uncertain graph $\mathcal{G}$ can be represented as a set $\{G_i = (V, E_{G_i})\}$ of size $2^{|E|}$. Each G_i within this set represents a possible world, corresponding to a specific realization where edges from the backbone graph are either present or absent based on a random selection process. Prior work typically establishes that the existence probabilities of the edges are independent [7, 13, 41, 46]. The probability $\mathbb{P}(G_i)$ of a given possible world G_i can be calculated using (1):

$$\mathbb{P}(G_i) = \prod_{e \in E_{G_i}} P(e) \prod_{e \in E \setminus E_{G_i}} (1 - P(e)) \tag{1}$$

For illustration, Figure 1 (b), (c), and (d) show the probabilities of three distinct possible worlds derived from the uncertain graph $\mathcal{G}$, respectively. Their corresponding existence probabilities can be computed using Equation (1).

2.2 Motif Counting

Motif counting in deterministic graphs is anchored in the concept of subgraph isomorphism. Here, a motif $\mathcal{M} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ is usually a connected graph with a few vertices [44].

Definition 2.2 (Subgraph Isomorphism [41]). *Given a motif $\mathcal{M} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ and a deterministic graph $G = (V, E)$, a subgraph isomorphism is an **injective function** f mapping $V_{\mathcal{M}}$ to V such that for every edge $e(u, v) \in E_{\mathcal{M}}$, there exists a corresponding edge $e(f(u), f(v)) \in E$.*

Given a motif $\mathcal{M}$ and a deterministic graph G, the task of motif counting is to ascertain the number of subgraph isomorphisms from $\mathcal{M}$ to G. In contrast, when counting motifs in an uncertain graph $\mathcal{G}$, the motif count becomes a **random variable**, stemming from the fact that each possible world G_i exists with an associated probability [41].

Problem Definition: For a given motif $\mathcal{M}$ and uncertain graph $\mathcal{G}$, the objective is to estimate both the **mean** and **variance** of the motif counts from $\mathcal{M}$ to $\mathcal{G}$.

Attempting a direct enumeration of all mappings from $\mathcal{M}$ to $\mathcal{G}$ is considered #P-hard [41]. This complexity arises from the need to count motif $\mathcal{M}$ across an exponentially large set of deterministic graphs that represent all possible worlds of $\mathcal{G}$. Even though prior research [41, 54, 55] has proposed several methods to pare down the computational time complexity, these approaches remain computationally intensive, particularly when applied to large graphs.

In the context of this paper, we characterize a motif as an *induced subgraph* in the backbone graph G_{bb} of the uncertain graph $\mathcal{G}$. The induced subgraph is defined as:

Definition 2.3 (Induced Subgraph [41]). *Given a deterministic graph $G = (V, E)$ and a vertex set $V' \subseteq V$, an induced subgraph $G_{sub} = (V', E')$ is the graph whose vertex set is V' and whose edge set E' consists of all the edges in E that have both endpoints in V'.*

Regarding non-induced motifs, they can be derived from induced motifs via a transformation, as elaborated in [30]. While our proposed methodology is capable of managing motif homomorphism counting, we predominantly focus on isomorphism counting. Consistent with traditional uncertain graph motif counting methods [41, 54], our experiments center on undirected uncertain graphs, emphasizing the structural nuances shaped by edge probabilities.

3 Subgraph Extraction and Refinement

In motif counting, the structural intricacies of both the motif and the target graph are crucial. However, given the often vast disparity between the target graph and the motifs, accurately correlating their representations becomes challenging.

ALSS [67] uses pre-trained models such as Node2Vec [22] or ProNE [64] to embed the topological information of the graph. However, this approach has two drawbacks. First, the embedding remains static throughout the training process. Second, these pre-trained models tend to capture broader global features, often neglecting the more relevant local topological information vital for motif counting.

Addressing these challenges, our model incorporates a subgraph extraction module, emphasizing local structures. Subgraphs are primarily extracted by centering on each node or edge (Section 3.1), followed by a learning-driven refinement to optimize these subgraphs for the most representative possible worlds (Section 3.2). The edge existence probability, denoted as P, is treated as the edge feature and plays an integral role in the subsequent refinement processes.

3.1 Subgraph Extraction

To extract subgraphs, a straightforward solution is to extract the ego net for each node.

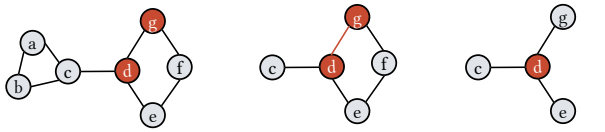
Definition 3.1 (k-hop Ego Net [17]). *Given a graph $G = (V, E)$, the k-hop ego net $G^k_{ego}[v] = (V^k_{ego}[v], E^k_{ego}[v])$ centered at node v is a subgraph of G whose vertex set $V^k_{ego}[v]$ includes v and all nodes reachable from v within k hops in G, and edge set $E^k_{ego}[v]$ includes all edges between nodes in $V^k_{ego}[v]$.*

Based on Definition 3.1, we can easily extract $|V|$ k-hop ego nets and pass them to the refinement module. Here, the hop number k is an important parameter to control the structure information within the subgraph. However, k + 1-hop ego nets may contain redundant structure information, while k-hop ego nets could miss crucial information, according to our experiments in Section 5.2.

To provide finer granularity, we propose the k-hop edge-ego net.

Definition 3.2 (k-hop Edge-Ego Net). *Given a graph $G = (V, E)$, the k-hop edge-ego net centered at edge (v, u), termed $G^k_{ego}[v, u] = (V^k_{ego}[v, u], E^k_{ego}[v, u])$, is a subgraph of G whose vertex set $V^k_{ego}[v, u]$ includes v and u, as well as all nodes reachable from v or u within k hops in G, and edge set $E^k_{ego}[v, u]$ includes all edges between nodes in $V^k_{ego}[v, u]$.*

Examining Definitions 3.1 and 3.2, we can find that $V^k_{ego}[v] \subseteq V^k_{ego}[v, u] \subseteq V^{k+1}_{ego}[v]$ for any edge (v, u), which is further illustrated in Fig. 2.



(a) Graph G (b) Edge-Ego Subgraph $G^1_{ego}[d, g]$ (c) Ego Subgraph $G^1_{ego}[d]$

Figure 2: Subgraph examples extracted by different methods.

Example 3.1. *As shown in Figure 2, the (edge) ego nets have nested relationships, i.e., $V^1_{ego}[d] \subseteq V^1_{ego}[d, g] \subseteq V^2_{ego}[d]$. Compared to $G^1_{ego}[d]$, $G^1_{ego}[d, g]$ can capture the 2-hop neighbor relationship between vertex d and f. Note that$v V^2_{ego}[d] = V_G$. Broadening the scope, $G^2_{ego}[d]$ introduces more vertices than $G^1_{ego}[d, g]$, including a and b.*

Collectively, ego nets and edge-ego nets offer fine-grained scopes of structural information for extracted subgraphs. In our algorithm, we consider the selection of the subgraph scope as a hyperparameter. Empirical results from our experiments reveal that 1-hop edge-ego nets deliver the most effective performance in motif counting tasks.

When dealing with graphs with high density, ego net extraction may produce graphs with high similarity, but it preserves the most important edges in the backbone graph compared with random walk-based subgraph extraction. The empirical evaluation in Section 5.3 demonstrates the superiority of our extraction method to random walk.

3.2 Uncertain Subgraph Structure Learning

In the previous step, the extracted subgraphs originate from the backbone graph, and they do not account for edge probabilities. The goal of this phase is to pinpoint the most representative possible world for each subgraph. When collectively considered, these selected possible worlds should reflect the intrinsic uncertainty of the entire uncertain graph, given that the same edge can exhibit varied statuses across different possible worlds depending on the associated subgraphs.

To this end, we introduce the **U**ncertain **S**ubgraph **S**tructure **L**earning (USSL) method. USSL is a learning-driven approach designed to emphasize vital edges while filtering out superfluous, "noisy" edges. This optimizes the representation of the uncertain subgraph, $\mathcal{G}_{ego}$. As a structure learning layer, USSL can be refined based on counting errors during back propagation.

To extract representative possible worlds via USSL, a key problem is which information should be included to gauge the importance of an edge. First, edge existence probability $P(\cdot)$ should be included, as it has an impact on the local structure. Second, the frequency of the edge $freq(\cdot)$, i.e., the number of extracted subgraphs containing the edge, is also important. A high frequency of edge occurrence signifies a large number of subgraphs requiring this edge for their construction, representing the edge is important to capture the correlation among different subgraphs and possible worlds.

Based on the above discussion, for each edge-ego net $G_{ego} = (V_{ego}, E_{ego})$, with a multi-layer perceptron, the edge (i, j)'s score can be calculated as:

$$\mathbf{Z}(i, j) = MLP([freq(i, j)||P(i, j)]), \tag{2}$$

where $[\cdot||\cdot]$ denotes the concatenation operation. To make the edge score comparable easily, the softmax function is applied for normalization:

$$\mathbf{Z}(i, j) = \frac{exp(\mathbf{Z}(i, j))}{\sum_{m=1}^{n} exp(\mathbf{Z}(i, m))}. \tag{3}$$

Leveraging the edge scores obtained from Equation (3), we can select edges with high scores to construct the refined subgraph G_{ego} as the selected possible world. In other words, USSL does not enumerate all the possible worlds for each subgraph, but constructs one representative possible world with the selected edges for each subgraph.

We suggest two edge-selection strategies: a ratio-centric approach and a score threshold method. In the former, USSL sorts edge scores in descending order, preserving the top ρ% edges. Conversely, the threshold approach retains edges with scores at least equal to the designated threshold ψ.

4 Representation Learning and Count Estimation

Upon deriving the possible worlds for the extracted subgraphs, we propose a hierarchical model tailored to capture the representation of the input uncertain graph $\mathcal{G}$. Meanwhile, we utilize our Motif Neural Network to extract the representation of the input motif $\mathcal{M}$. Armed with these dual representations, a specialized counting unit is integrated to adeptly estimate the motif counts.

4.1 Target Graph Neural Network

To learn the representation of the uncertain graph $\mathcal{G}$ from the refined subgraphs, we employ a hierarchical graph neural network. Our model for the target graph is demonstrated in Fig. 3. For each subgraph, our model learns the vertex representation obtained from USSL via component GNN and then aggregates the vertex representation by a subgraph pooling layer to readout the subgraph representation. In line with prior research [65], we incorporate a graph pooling layer into our methodology. This layer plays a crucial role in aggregating the representations of all the subgraphs,

ultimately allowing us to derive a comprehensive representation of the entire graph. As shown in Fig. 3, the component GNN, subgraph readout, and graph readout collectively form the hierarchical structure of our target graph neural network.

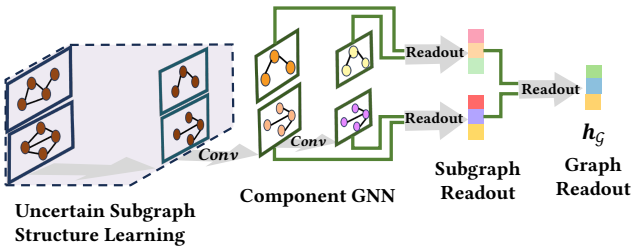


Figure 3: The architecture of target graph neural network.

Component GNN. The primary objective of the Component GNN is to acquire the initial representations for individual vertices within a given subgraph through iterative message passing [20]. Subsequently, these vertex representations are summarized and aggregated using a pooling layer to derive the overall subgraph representation. This mechanism allows us to capture and encode the structural and contextual information of the subgraph efficiently.

Component GNN is not limited to a specific GNN model. Taking vertex i as example, to calculate its hidden state $\mathbf{h}_i^{(k)} \in \mathbb{R}^d$ at the time step k, the component GNN can be expressed as:

$$\mathbf{h}_i^{(k)} = \zeta^{(k)}(\mathbf{h}_i^{(k-1)}, \mathbf{m}_i^{(k)}), \tag{4}$$

$$\mathbf{m}_i^{(k)} = Agg_{j \in N(i)}(\eta^{(k)}(\mathbf{h}_i^{(k-1)}, \mathbf{h}_j^{(k-1)}, \mathcal{E}_{ij})), \tag{5}$$

where $\zeta^{(k)}(\cdot)$ denotes the update function and $\eta^{(k)}(\cdot)$ denotes the message function. The feature vector $\mathbf{x}_i$ obtained from USSL serves as the initial state $\mathbf{h}_i^0$. Hidden state $\mathbf{h}_i^{(k)}$ is updated by previous state $\mathbf{h}_i^{(k)}$ at time step k and the message $\mathbf{m}_i^{(k)}$ from its neighbors. The message $\mathbf{m}_i^{(k)}$ at time step k is calculated by (5) where $\mathcal{E}_{ij}$ denotes the edge features of edge (i, j) and $Agg(\cdot)$ denotes the permutation invariant aggregation function such as $Sum(\cdot)$, $Mean(\cdot)$ or $Max(\cdot)$. It is worth noticing that the component GNN shares the parameters and it is applied on each subgraph in subgraph set S_{sub}.

Readout. After K time steps, we get K hidden states of each vertex. To fully explore the structure information in different feature spaces, the hidden states of each time step $k \in [1, K]$ are concatenated together as the final vertex representation:

$$\mathbf{h}_i = [\mathbf{h}_i^{(1)} || \ldots || \mathbf{h}_i^{(K)}]. \tag{6}$$

A pooling layer is applied to aggregate the vertex representation within the subgraph and obtain the subgraph representation $\mathbf{h}_{G_{ego}}$, which can be expressed as:

$$\mathbf{h}_{G_{ego}} = Readout_0(\mathbf{h}_i \mid i \in V_{ego}), \tag{7}$$

where $Readout_0(\cdot)$ is the subgraph pooling layer to summarize the vertex representations in each subgraph.

To obtain the whole graph representation, we apply another graph pooling layer to summarize all the subgraph representations:

$$\mathbf{h}_{\mathcal{G}} = Readout_1(\mathbf{h}_{G_{ego}} \mid G_{ego} \in S_{sub}), \tag{8}$$

where $Readout_1(\cdot)$ is the graph pooling layer to aggregate the subgraph representation to the whole uncertain graph representation.

Comparison with MPNN. Compared with vanilla message-passing neural networks such as GCN [33], GAT [57] and GIN [59], our model focuses on learning vertex representations within individual subgraphs rather than at the whole graph level. This approach effectively scales down the receptive field from the entire graph to the subgraph level. Subsequently, we obtain the overall graph representation through a hierarchical process involving two pooling layers. When coupled with USSL, which teases out possible worlds from ego nets, our hierarchical approach adeptly integrates insights at both subgraph and graph levels, maintaining the uncertain nature of the target graph. Crucially, in different subgraphs, the vertex i will have different hidden states and messages during message passing.

4.2 Motif Neural Network

In Section 4.1, we detailed a strategy to represent the uncertain graph $\mathcal{G}$. For motifs, i.e., small connected graphs [44], this strategy is overkill. Classic GNNs can aptly represent the structural nuances of motifs.

While GCN [33], GAT [57], and GraphSAGE [24] are prominent GNNs with efficiency in node-centric tasks, they are limited in preserving graph structures, especially when confronted with the expressiveness challenges of the 1-Weisfeiler-Lehman (1-WL) test [59]. GIN [59] and its variant GINE [26], on the other hand, effectively address this limitation using multi-layer perceptrons $MLP(\cdot)$. Notably, GINE factors in edge features for enhanced performance.

Empirical results from our experiments reveal that GINE performs best among the classic GNN models. Hence, we employ GINE as the motif neural network to learn the representation of motif.

GINE involves edge features $\mathcal{E}$ into aggregation to update the hidden state $\mathbf{h}_i^{(k)}$ at time step k and applies $Relu(\cdot)$ as activation function:

$$\mathbf{h}_i^{(k)} = MLP((1 + \epsilon^{(k)})\mathbf{h}_i^{(k-1)} + \sum_{j \in N(i)} Relu(\mathbf{h}_j^{(k-1)} + \mathcal{E}_{ij})) \tag{9}$$

For the holistic representation, GINE employs a summation readout function, proven injective in [59], culminating in the final representation:

$$\mathbf{h}_{\mathcal{M}}^{(k)} = Sum(\{\mathbf{h}_i^{(k)} \mid i \in V_{\mathcal{M}}\}), \tag{10}$$

$$\mathbf{h}_{\mathcal{M}} = [\mathbf{h}_{\mathcal{M}}^{(1)} || \ldots || \mathbf{h}_{\mathcal{M}}^{(K)}] \tag{11}$$

4.3 Counting Unit

Leveraging the representations of motifs ($\mathbf{h}_{\mathcal{M}}$) and the uncertain graph ($\mathbf{h}_{\mathcal{G}}$), our goal is to estimate the mean and variance of motif M within graph $\mathcal{G}$. A straightforward strategy concatenates the representations and utilizes a multi-layer perceptron (MLP) to predict the mean and variance:

$$\hat{m}, \hat{v} = MLP([\mathbf{h}_{\mathcal{M}} || \mathbf{h}_{\mathcal{G}}]), \tag{12}$$

The objective function for optimizing model parameters is defined as:

$$\mathcal{L}_\Theta = \alpha \mathcal{L}(m, \hat{m}) + (1 - \alpha)\mathcal{L}(v, \hat{v}), \tag{13}$$

where α balances the weight between mean and variance, Θ denotes model parameters, and Huber Loss [28] is employed as the loss

function $\mathcal{L}(\cdot,\cdot)$, balancing robustness and penalization of large errors.

Nonetheless, this method cannot fully capture the intricate correlation between motif representations and graph representations. In motif counting, our objective is to infer and leverage the inherent correlations between motifs and the underlying graph structure. When a motif aligns with a portion of the input graph, we aim to extract and analyze the latent relationships by exploring the correlation between their respective representation vectors. In [61], Feature-wise Linear Modulation (FiLM) [47] is applied to fine-tune the graph representation only by a linear transformation to exploit the correlation between query graph and input graph, which can limit its capacity to capture more complex relationships in certain scenarios.

Inspired by FiLM, our model adopts Soft Canonical Correlation Analysis (Soft CCA) [11] to exploit the underlying correlations by non-linear transformation between input representations of motif and graph. Soft CCA [11] leverages Lagrangian Relaxation [36] to remove the hard decorrelation constraint in original CCA [53], which can be expressed as:

$$\begin{aligned}&\mathcal{L}_{dist}(MLP(\mathbf{h}_{\mathcal{M}}), MLP(\mathbf{h}_{\mathcal{G}}))\\ &+\lambda(\mathcal{L}_{dl}(MLP(\mathbf{h}_{\mathcal{M}})) + \mathcal{L}_{dl}(MLP(\mathbf{h}_{\mathcal{G}})))\end{aligned} \tag{14}$$

where $\lambda \geq 0$ is a trade-off hyper-parameter. $\mathcal{L}_{dist}$ denotes the correlation between two embedding vectors, which is calculated by

$$\mathcal{L}_{dist} = \frac{1}{2}\|MLP(\mathbf{h}_{\mathcal{M}}) - MLP(\mathbf{h}_{\mathcal{G}})\|_F^2. \tag{15}$$

$\mathcal{L}_{dl}$ denotes decorrelation loss. Following [63], the decorrelation loss is performed on normalized representation vector to constrain off-diagonal elements of covariance matrix close to 0:

$$\mathcal{L}_{dl}(U) = \|U^T U - I\|_F^2, \tag{16}$$

where U is the normalized representation vector and I is the identity matrix.

With the CCA Loss as the regularizer, our objective function can be expressed as:

$$\mathcal{L}_{\Theta}(\mathcal{M}) = \alpha\mathcal{L}(m,\hat{m}) + (1-\alpha)\mathcal{L}(v,\hat{v}) + \gamma\mathcal{L}_{CCA} \tag{17}$$

where CCA loss $\mathcal{L}_{CCA}$ is calculated by (14) and γ is a hyperparameter to adjust the importance of CCA loss. The theorem below gives the upper bound of estimated mean and variance value:

Theorem 1. *Assume that all the operations in MLP are locally Lipschitz-continuous and that their partial derivatives of non-linear activation function can be computed and efficiently maximized. The predicted mean and variance value's bounds are*

$$m - 2\tau \leq \hat{m} \leq \tau \tag{18}$$

$$v - 2\tau \leq \hat{v} \leq \tau \tag{19}$$

where $\tau \leq \prod_{k=1}^{K}\|W_k\|_2$, *and* W_k *is the weight matrix of MLP.*

The proof of Theorem 1 is given in the full version of this paper.

5 Experimental Studies

In this section, we present our experimental setting and experimental results. Due to page limit, dataset description, implementation details and several supplementary experiments are in the full version.

5.1 Experimental Setup

Dataset. We evaluate our method alongside state-of-the-art solutions using five real-world datasets, including one sensor information transition dataset(**ITL** [42]), one Protein-Protein Interaction (PPI) dataset (**KRC** [34]), one citation network (**DBLP** [1]) and two road networks (**BJ** [14] and **CAL** [37]). The dataset statistics are presented in Table 1, and the details of motifs are in Table 2.

Table 1: Statistics of datasets.

Dataset	|V|	|E|	Min/max/avg P(e)
ITL	55	1,342	0.01/0.83/0.17
KRC	2,708	7,123	0.27/0.99/0.68
DBLP	26,985	38,117	0.39/0.99/0.51
BJ	75,958	109,741	0.90/0.99/0.95
CAL	1,965,206	2,766,607	0.01/0.99/0.50

Table 2: Details of motifs.

Dataset	ITL	KRC	DBLP	BJ	CAL
#Motifs	543	833	807	607	653
Motif Sizes	{3,4}	{3,4,5,6}	{3,4,5,6}	{3,4}	{3,4}

LINC [41] is applied to generate the ground-truth labels for each motif as it proposes the latest solution for uncertain graph counting. LINC failed to give the ground truth pairs for some 5-node and 6-node motifs over ITL, BJ, and CAL datasets because either LINC cannot give the result in a reasonable time or the corresponding motif cannot be found from possible worlds. Therefore, only 3-node and 4-node motifs are selected in our experiment on the three datasets. The motifs are randomly sampled from the datasets. The details of query motifs are in Table 2.

Baseline Methods. To evaluate the accuracy of our proposed method, we compare it with the state-of-the-art learning-based method ALSS [67], EGNN [21], NeurSC [58] and heuristic method BMA [54]. For ALSS, We select ProNE [64] as the label embedding model and select the active learning strategy as *consist* according to the default setting in its implementation. We also combined LINC with ALSS (Sampler-ALSS): we use LINC to sample 100 possible worlds for the uncertain graph, run ALSS on sampled possible worlds and obtain the average results. To achieve the best results on our dataset, we adjust ALSS's parameters referring to the tuning range in [67]. The source code of EGNN is obtained from the author, and the best parameters on each dataset are obtained by cross-validation. For efficiency, we further compare our solution with LINC [41] and BMA [54].

Evaluation Metrics. In our experiments, we evaluate the methods by both effectiveness and efficiency. For effectiveness evaluation, we employ mean absolute percentage error (MAPE) [15] and Q-error [45]. Mean absolute percentage error is defined as:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right|, \tag{20}$$

where $\hat{y}_i$ denotes the predicted value and y_i denotes the ground-truth value obtained from LINC. Q-error is defined as:

$$\text{Q-error} = \max\left(\frac{\hat{y}_i}{y_i}, \frac{y_i}{\hat{y}_i}\right) \tag{21}$$

Both metrics exhibit better performance as their values decrease. To evaluate the efficiency, we compared the average query processing time with ALSS [67] and LINC [41]. For tables reporting accuracy results (e.g., Tables 5 and 6), mean absolute percentage errors are in percentages, with the optimal results **bolded** and the second-best results underlined.

5.2 Accuracy Comparison

In this subsection, we compare the motif counting accuracy of our proposed model, UnG-MoCha, against the baselines. Furthermore, ablation studies are conducted to investigate the influence of various UnG-MoCha modules on its overall performance.

The comparison results between baselines and our UnG-MoCha are illustrated in Fig. 4 by box-plot. Due to limited space, we report the results based on MAPE. The results based on Q-Error is similar to the performance based on MAPE. The results indicate that UnG-MoCha generally outperforms the baselines across various datasets and motif sizes. Despite ALSS showing a smaller MAPE in counting 4-node motifs on the BJ dataset and a smaller MAPE on the CAL dataset for 4-node motifs in variance value prediction, UnG-MoCha consistently exhibits lower median MAPE, suggesting superior and stable performance across diverse instances. Although the results produced by BMA show smaller lower bounds when counting 3-node motifs on DBLP and BJ datasets in mean value prediction, the median value is almost the same as it of UnG-MoCha and BMA shows a quite large difference between the upper and lower bounds.

To further investigate why baselines perform slightly better on BJ and CAL, we observe that these datasets, which symbolize real-world road networks, rarely contain dense 4-node motifs like 4-cliques. This unique characteristic somewhat mitigates the limitation of ALSS: its query graph decomposition strategy constructs l-hop BFS trees to extract substructures from each vertex in query graphs, which may miss dense substructures when l is small. Additionally, ALSS focuses solely on learning query graph representations, potentially overlooking valuable information present in the target graph. EGNN treats edge probabilities as normal features instead of existence probability and cannot capture the semantics of possible worlds well, which are captured by USSL in our model. Further, it focuses on global topological information, and its embedding contains less local topological information than our hierarchical solution, resulting in poor accuracy. Sampler-ALSS has the same drawbacks as ALSS, and it also suffers from high time complexity because it needs to perform on all the sampled possible worlds.

In contrast, UnG-MoCha proficiently learns representations of both target graphs and motifs, while employing USSL to refine the subgraphs extracted from the target graph. The experimental results attest to our method's enhanced capability to adeptly navigate motif counting tasks over uncertain graphs.

5.3 Ablation Studies

Subgraph Refinement Strategy Comparison. To demonstrate the effectiveness of our subgraph refinement strategy USSL, Table 3 reports the performance comparison between leveraging USSL and directly using the subgraphs extracted from 1-hop edge-ego net extraction. As shown in Table 3, with USSL, UnG-MoCha achieves a much more accurate result than directly performing on the extracted subgraphs. The results indicate that the USSL improves the prediction performance on most datasets.

Subgraph Extraction Comparison. Table 5 presents the performance comparison between ego net extraction, edge-ego net extraction (Section 3.1) and random walk. The random walk-based method first simulates 20 independent random walkers taking 20 steps for each vertex i in the subgraph to generate 20 possible worlds. Among the 20 possible worlds, the one with the highest probability will be selected as the representative subgraph. The findings demonstrate the effectiveness of both extraction strategies in motif counting tasks involving uncertain graphs. Furthermore, edge-ego net extraction surpasses ego net extraction in terms of effectiveness. The results also indicate that even when dealing with graphs with high density, ego net extraction can preserve the most important edges and improve performance with the help of USSL on high-density datasets. As discussed in Section 3.1, with consistent trends observed across other datasets, 1-hop edge-ego nets excel in capturing edges not present in 1-hop ego nets while avoiding redundancy found in 2-hop or larger k-hop ego nets. Hence, the 1-hop edge-ego net is chosen as the default subgraph extraction strategy.

Motif Neural Network Comparison. We evaluate various GNNs for motif representation learning, highlighting results from the KRC dataset in Table 6, with consistent trends observed across other datasets. GINE excels in predicting both mean and variance values, outperforming other GNN models. The enhanced performance of GIN and GINE, utilizing $MLP(\cdot)$ for aggregation and combination, underscores their elevated expressive capability, where GINE further incorporates edge features.

Counting Unit Comparison. We explore the impact of different counting units by comparing CCANet with DIAMNet [39] and FilMNet [61], as shown in Table 4. CCANet predominates in both mean and variance value prediction across most datasets. DIAMNet struggles with larger graphs because it depends on the dynamic intermedium attention memory mechanism to preserve the graph information, which is limited by memory size and can only leverage partial data graph information. FilMNet, albeit superior in variance value prediction on the BJ dataset, is hampered by its linear transformation focus and inability to discern correlations between graph and motif representations in a non-linear transformed latent space.

5.4 Efficiency Comparison

Average querying time comparison. To assess efficiency, we compare the average query processing times of our method and baselines across various datasets and motif sizes, as depicted in Fig. 5. Sampler-ALSS's training and inference time is related to vanilla ALSS, which is the time cost of ALSS times the number of sampled possible worlds. Therefore, we focus on vanilla ALSS to analyze its efficiency. While ALSS, EGNN, and our method outperform LINC and BMA in all contexts, our method exhibits minimal sensitivity to motif size compared to ALSS, especially for 5-node and 6-node

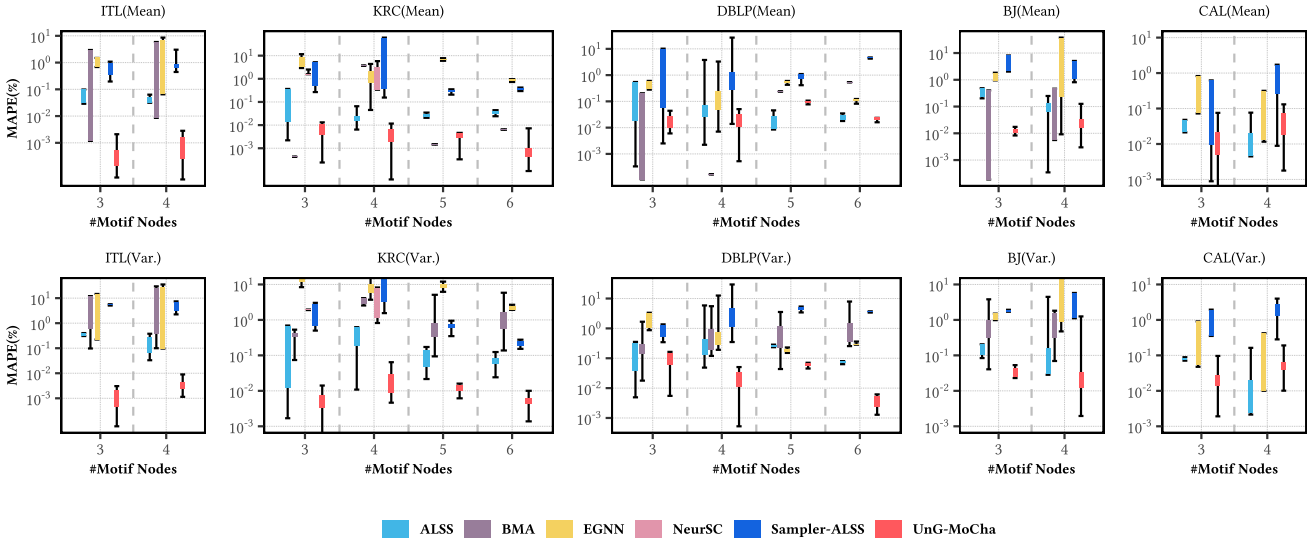


Figure 4: Mean absolute percentage error comparison.

Table 3: Accuracy evaluation of mean over different refinement strategies.

Methods	ITL		KRC		DBLP		BJ		CAL	
	MAPE	Q-error	MAPE	Q-error	MAPE	Q-error	MAPE	Q-error	MAPE	Q-error
non-USSL	0.459±0.080	1.005±0.008	5.598±0.305	1.060±0.035	**1.054**±2.297	1.056±1.296	1.320±0.068	1.013±0.007	6.827±0.507	1.135±0.052
USSL	**0.076**±0.087	**1.001**±0.001	**0.485**±0.370	**1.000**±0.001	1.083±0.878	**1.001**±0.001	**0.552**±0.308	**1.001**±0.001	**3.970**±3.482	**1.004**±0.003

Table 4: Accuracy evaluation of mean value over different counting units.

Methods	ITL		KRC		DBLP		BJ		CAL	
	MAPE	Q-error	MAPE	Q-error	MAPE	Q-error	MAPE	Q-error	MAPE	Q-error
FiLMNet	49.547±122.08	1.190±0393	**1.982**±0.978	1.012±0.002	13.616±7.433	1.015±0.009	**2.416**±1.586	**1.003**±0.002	5.209±3.815	1.006±0.004
DIAMNet	18.760±28.945	1.153±0.345	38.356±9.112	1.060±0.041	3.969±1.217	**1.005**±0.004	36.095±25.323	1.364±0.611	34.349±9.596	1.041±0.011
CCANet	**11.471**±30.637	**1.108**±0.310	2.568±0.901	**1.003**±0.015	**2.446**±1.810	1.008±0.013	4.556±7.004	1.007±0.012	**3.970**±3.482	**1.004**±0.003

Table 5: Subgraph extraction strategy comparison on ITL.

Methods	MAPE(Mean)	MAPE(Var.)	Q-Error(Mean)	Q-Error(Var.)
1-hop ego	0.522±0.176	1.829±1.104	1.005±0.001	1.018±0.015
1-hop edge-ego	**0.085**±0.063	0.272±0.254	**1.001**±0.001	**1.003**±0.002
2-hop ego	0.931±0.676	2.104±1.147	1.009±0.007	1.021±0.019
2-hop edge-ego	0.582±0.289	1.380±0.116	1.006±0.001	1.014±0.005
3-hop ego	2.493±0.167	4.501±0.310	1.025±0.017	1.046±0.036
3-hop edge-ego	0.444±0.033	1.480±0.152	1.004±0.003	1.015±0.016
1-hop BFS	1.511±0.11	3.025±0.31	1.013±0.010	1.031±0.025
random walk	0.692±0.231	**0.038**±0.026	1.007±0.002	1.039±0.014

Table 6: Motif neural network comparison on KRC.

Methods	MAPE(Mean)	MAPE(Var.)	Q-Error(Mean)	Q-Error(Var.)
GCN	16.742±56.670	95.078±287.284	1.252±0.358	1.323±0.320
GAT	18.307±58.775	123.268±372.917	1.250±0.363	1.317±0.334
SAGE	22.938±21.301	110.611±163.684	1.023±0.019	1.053±0.047
GIN	14.913±17.776	49.575±41.009	1.016±0.022	1.043±0.052
GINE	**0.485**±0.370	**1.421**±1.656	**1.000**±0.001	**1.001**±0.001

motifs across most datasets, and provides up to 6 orders of magnitude speedup over LINC. This is attributable to ALSS's quadratic time growth due to its query graph decomposition strategy. Despite its efficiency, particularly with smaller motifs on larger data graphs, ALSS's method—embedding only substructures from the query graph—compromises its accuracy, as discussed in Section 5.2. Our model also outperforms EGNN on most datasets. Besides, EGNN applies doubly stochastic normalization, which brings $O(|E| \times |E|)$ extra time cost compared with vanilla GNN models. BMA needs to sample more possible worlds than LINC to obtain accurate estimation results, which causes high time costs.

5.5 Inductive Test

We also evaluated the inductive learning performance of UnG-MoCha. Firstly, we pre-train our model on KRC dataset with only small size motifs as training data, then fine-tune the pre-trained model with different number of training data under three settings: predict the count of large size motifs on KRC, it of small size motifs on DBLP and large size motifs on DBLP. The result is visualized in Figure 6. With the increase of training data, the pre-trained model achieves better performance. With 20% of training data, the

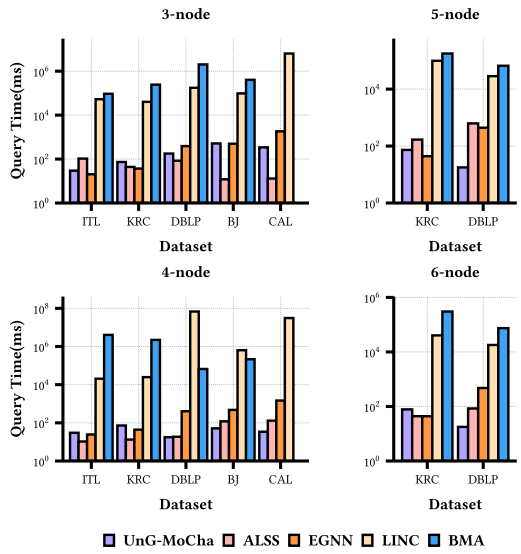


Figure 5: Average query processing time (ms).

fine-tuned model can achieve acceptable results. It is interesting that the pre-trained model achieves a relatively lower MAPE when estimating the counts of small-size motifs on an unseen dataset, demonstrating UnG-MoCha is less impacted by the input graph. The result indicates that UnG-MoCha can be applied to handle large-scale graphs after pre-trained on smaller-size datasets.

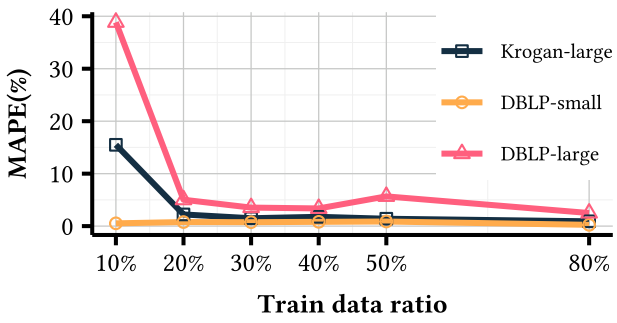


Figure 6: Results of inductive test.

6 Related Work

Graph Representation Learning. Graph representation learning is a kind of technique that encodes the nodes in the complex network into low-dimensional vectors and maintains the graph structure information as much as possible. Graph neural network (GNN) models [24, 33, 57, 59, 65] learn the graph representation via deep learning and show great success in recent years. For example, GIN [59] shows that GNN models can have the same discrimination ability as Weisfeiler-Leman (1-WL) test [35]. Contrasting traditional embedding methods, GNNs offer the notable capability to generalize to unseen nodes and new graphs, providing pertinent representations. Most GNN models follow the idea of convolutional neural networks and message-passing mechanisms [20].

Graph Structure Learning. Graph structure learning (GSL) [62, 66, 68] has attracted substantial attention in recent years and proved its efficiency in learning robust graph representation. However, GSL has not been well studied on uncertain graphs. There is no learning-based method for uncertain graphs, and the existing method [46] is an unsupervised method, lacking optimization in accordance with errors from downstream tasks.

Subgraph Matching and counting. There are two main categories of subgraph matching: backtracking approach [10, 32, 51, 56] and join-based approach [2, 52]. Through exhaustive enumeration of all potential subgraph isomorphisms, these methods are able to derive exact counts for subgraphs.

For subgraph counting, traditional algorithms [9, 18, 25, 41, 54] suffer from high time complexity. Learning-based subgraph isomorphism counting methods [39, 40, 58, 61, 67] attempt to find the approximate isomorphic subgraph counts with lower time complexity. NSIC [39] first proposed GNN-based subgraph isomorphism counting and leverages the attention memory mechanism to reduce the time complexity. ALSS [67] combines active learning with the GNN model to count subgraph isomorphism or homomorphism. However, there is no ML method focusing on subgraph isomorphism counting on uncertain graphs.

7 Conclusion

In this paper, we introduce UnG-MoCha, a novel learning-based approach designed to estimate the mean and variance values of motif counts in uncertain graphs. Our method initially leverages uncertain subgraph structure learning on extracted subgraphs, capturing edge uncertainty effectively. Subsequently, target graph neural network is employed to learn the representation of the uncertain graph. Simultaneously, motif representation is learnt by motif neural network. Through the incorporation of the counting unit, our approach adeptly mines correlations between motif and graph representations, enhancing estimation accuracy. Extensive experiments demonstrate the effectiveness and efficiency of UnG-MoCha in motif counting tasks on uncertain graphs, highlighting its potential utility and applicability in further graph analysis scenarios. For future work, we plan to study how to extend our solution to support larger-size motifs.

Acknowledgments

Chenhao Ma was partially supported by NSFC under Grant 62302421, Basic and Applied Basic Research Fund in Guangdong Province under Grant 2023A1515011280, 2025A1515010439, Ant Group through CCF-Ant Research Fund, Shenzhen Research Institute of Big Data under grant SIF20240004, and the Guangdong Provincial Key Laboratory of Big Data Computing, The Chinese University of Hong Kong, Shenzhen. Xiaolin Han was supported by NSFC under Grant 62302397, the fund of Laboratory for Advanced Computing and Intelligence Engineering (No. 2023-LYJJ-01-021).

References

[1] 2023. DBLP dataset. https://dblp.uni-trier.de/db/

[2] Christopher R Aberger, Andrew Lamb, Susan Tu, Andres Nötzli, Kunle Olukotun, and Christopher Ré. 2017. Emptyheaded: A relational engine for graph processing. *ACM Transactions on Database Systems (TODS)* 42, 4 (2017), 1–44.

[3] Serge Abiteboul, Paris Kanellakis, and Gosta Grahne. 1987. On the Representation and Querying of Sets of Possible Worlds. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data* (San Francisco, California, USA) *(SIGMOD '87)*. Association for Computing Machinery, New York, NY, USA, 34–48. doi:10.1145/38713.38724

[4] Nesreen K Ahmed, Jennifer Neville, Ryan A Rossi, and Nick Duffield. 2015. Efficient graphlet counting for large networks. In *2015 IEEE international conference on data mining*. IEEE, 1–10.

[5] Saurabh Asthana, Oliver D King, Francis D Gibbons, and Frederick P Roth. 2004. Predicting protein complex membership using probabilistic network reliability. *Genome research* 14, 6 (2004), 1170–1175.

[6] Manuel Bodirsky. 2015. Graph homomorphisms and universal algebra course notes. *TU Dresden* (2015).

[7] Paolo Boldi, Francesco Bonchi, Aris Gionis, and Tamir Tassa. 2012. Injecting uncertainty in graphs for identity obfuscation. *arXiv preprint arXiv:1208.4145* (2012).

[8] Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. 2017. Counting graphlets: Space vs time. In *Proceedings of the tenth ACM international conference on web search and data mining*. 557–566.

[9] Vincenzo Carletti, Pasquale Foggia, Alessia Saggese, and Mario Vento. 2017. Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with VF3. *IEEE transactions on pattern analysis and machine intelligence* 40, 4 (2017), 804–818.

[10] Vincenzo Carletti, Pasquale Foggia, and Mario Vento. 2015. VF2 Plus: An improved version of VF2 for biological graphs. In *Graph-Based Representations in Pattern Recognition: 10th IAPR-TC-15 International Workshop, GbRPR 2015, Beijing, China, May 13-15, 2015. Proceedings 10*. Springer, 168–177.

[11] Xiaobin Chang, Tao Xiang, and Timothy M Hospedales. 2018. Scalable and effective deep CCA via soft decorrelation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1488–1497.

[12] Xiaowei Chen and John CS Lui. 2018. Mining graphlet counts in online social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 12, 4 (2018), 1–38.

[13] Yifan Chen, Xiang Zhao, Xuemin Lin, and Yang Wang. 2015. Towards frequent subgraph mining on single large uncertain graphs. In *2015 IEEE International Conference on Data Mining*. IEEE, 41–50.

[14] OpenStreetMap Contributors. 2017. Roadnet Data. https://www.openstreetmap.org

[15] Arnaud De Myttenaere, Boris Golden, Bénédicte Le Grand, and Fabrice Rossi. 2016. Mean absolute percentage error for regression models. *Neurocomputing* 192 (2016), 38–48.

[16] Ove Frank. 1979. Estimating a graph from triad counts. *Journal of Statistical Computation and Simulation* 9, 1 (1979), 31–46.

[17] Linton C Freeman. 1982. Centered graphs and the structure of ego networks. *Mathematical Social Sciences* 3, 3 (1982), 291–304.

[18] Takayasu Fushimi, Kazumi Saito, and Hiroshi Motoda. 2022. Efficient computation of expected motif frequency in uncertain graphs by exploiting possible world marginalization and motif transition. *Social Network Analysis and Mining* 12, 1 (2022), 126.

[19] Joy Ghosh, Hung Q Ngo, Seokhoon Yoon, and Chunming Qiao. 2007. On a routing problem within probabilistic graphs and its application to intermittently connected networks. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE, 1721–1729.

[20] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.

[21] Liyu Gong and Qiang Cheng. 2019. Exploiting edge features for graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9211–9219.

[22] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. arXiv:1607.00653 [cs.SI]

[23] Saket Gurukar, Sayan Ranu, and Balaraman Ravindran. 2015. Commit: A scalable approach to mining communication motifs from dynamic networks. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*. 475–489.

[24] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[25] Wook-Shin Han, Jinsoo Lee, and Jeong-Hoon Lee. 2013. Turboiso: towards ultrafast and robust subgraph isomorphism search in large graph databases. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. 337–348.

[26] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2019. Strategies for pre-training graph neural networks. *arXiv preprint arXiv:1905.12265* (2019).

[27] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. 2014. Querying k-truss community in large and dynamic graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1311–1322.

[28] Peter J Huber. 1992. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*. Springer, 492–518.

[29] Mark Jerrum and Kitty Meeks. 2015. The parameterised complexity of counting connected subgraphs and graph motifs. *J. Comput. System Sci.* 81, 4 (2015), 702–716.

[30] Madhav Jha, C Seshadhri, and Ali Pinar. 2015. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *Proceedings of the 24th international conference on world wide web*. 495–505.

[31] Rui Jiang, Zhidong Tu, Ting Chen, and Fengzhu Sun. 2006. Network motif identification in stochastic networks. *Proceedings of the National Academy of Sciences* 103, 25 (2006), 9404–9409.

[32] Hyunjoon Kim, Yunyoung Choi, Kunsoo Park, Xuemin Lin, Seok-Hee Hong, and Wook-Shin Han. 2023. Fast subgraph query processing and subgraph matching via static and dynamic equivalences. *The VLDB journal* 32, 2 (2023), 343–368.

[33] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[34] Nevan J Krogan, Gerard Cagney, Haiyuan Yu, Gouqing Zhong, Xinghua Guo, Alexandr Ignatchenko, Joyce Li, Shuye Pu, Nira Datta, Aaron P Tikuisis, et al. 2006. Global landscape of protein complexes in the yeast Saccharomyces cerevisiae. *Nature* 440, 7084 (2006), 637–643.

[35] AA Leman and Boris Weisfeiler. 1968. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya* 2, 9 (1968), 12–16.

[36] Claude Lemaréchal. 2001. Lagrangian relaxation. *Computational combinatorial optimization: optimal or provably near-optimal solutions* (2001), 112–156.

[37] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. 2009. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics* 6, 1 (2009), 29–123.

[38] Xiaodong Li, Reynold Cheng, Matin Najafi, Kevin Chang, Xiaolin Han, and Hongtai Cao. 2020. M-cypher: A gql framework supporting motifs. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 3433–3436.

[39] Xin Liu, Haojie Pan, Mutian He, Yangqiu Song, Xin Jiang, and Lifeng Shang. 2020. Neural subgraph isomorphism counting. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1959–1969.

[40] Xin Liu and Yangqiu Song. 2022. Graph convolutional networks with dual message passing for subgraph isomorphism counting and matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 7594–7602.

[41] Chenhao Ma, Reynold Cheng, Laks VS Lakshmanan, Tobias Grubenmann, Yixiang Fang, and Xiaodong Li. 2019. Linc: a motif counting algorithm for uncertain graphs. *Proceedings of the VLDB Endowment* 13, 2 (2019), 155–168.

[42] Samuel Madden. 2004. Intel lab data. http://db.csail.mit.edu/labdata/labdata.html

[43] Ine Melckenbeeck, Pieter Audenaert, Didier Colle, and Mario Pickavet. 2018. Efficiently counting all orbits of graphlets of any order in a graph using autogenerated equations. *Bioinformatics* 34, 8 (2018), 1372–1380.

[44] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.

[45] Guido Moerkotte, Thomas Neumann, and Gabriele Steidl. 2009. Preventing bad plans by bounding the impact of cardinality estimation errors. *Proceedings of the VLDB Endowment* 2, 1 (2009), 982–993.

[46] Panos Parchas, Nikolaos Papailiou, Dimitris Papadias, and Francesco Bonchi. 2018. Uncertain graph sparsification. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (2018), 2435–2449.

[47] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[48] Ali Pinar, Comandur Seshadhri, and Vaidyanathan Vishal. 2017. Escape: Efficiently counting all 5-vertex subgraphs. In *Proceedings of the 26th international conference on world wide web*. 1431–1440.

[49] Tanay Kumar Saha and Mohammad Al Hasan. 2015. Finding network motifs using MCMC sampling. In *Complex Networks VI: Proceedings of the 6th Workshop on Complex Networks CompleNet 2015*. Springer, 13–24.

[50] Erich Schubert, Alexander Koos, Tobias Emrich, Andreas Züfle, Klaus Arthur Schmid, and Arthur Zimek. 2015. A framework for clustering uncertain data. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1976–1979.

[51] Shixuan Sun, Xibo Sun, Bingsheng He, and Qiong Luo. 2022. RapidFlow: an efficient approach to continuous subgraph matching. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2415–2427.

[52] Zhao Sun, Hongzhi Wang, Haixun Wang, Bin Shao, and Jianzhong Li. 2012. Efficient subgraph matching on billion node graphs. *arXiv preprint arXiv:1205.6691* (2012).

[53] Bruce Thompson. 1984. *Canonical correlation analysis: Uses and interpretation*. Number 47. Sage.
[54] Andrei Todor, Alin Dobra, and Tamer Kahveci. 2015. Counting Motifs in Probabilistic Biological Networks. In *Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics* (Atlanta, Georgia) *(BCB '15)*. Association for Computing Machinery, New York, NY, USA, 116–125. doi:10.1145/2808719.2808731
[55] Ngoc Hieu Tran, Kwok Pui Choi, and Louxin Zhang. 2013. Counting motifs in the human interactome. *Nature communications* 4, 1 (2013), 2241.
[56] J. R. Ullmann. 1976. An Algorithm for Subgraph Isomorphism. *J. ACM* 23, 1 (jan 1976), 31–42. doi:10.1145/321921.321925
[57] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[58] Hanchen Wang, Rong Hu, Ying Zhang, Lu Qin, Wei Wang, and Wenjie Zhang. 2022. Neural subgraph counting with wasserstein estimator. In *Proceedings of the 2022 International Conference on Management of Data*. 160–175.
[59] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).
[60] Soon-Hyung Yook, Zoltán N Oltvai, and Albert-László Barabási. 2004. Functional and topological characterization of protein interaction networks. *Proteomics* 4, 4 (2004), 928–942.
[61] Xingtong Yu, Zemin Liu, Yuan Fang, and Xinming Zhang. 2023. Learning to count isomorphisms with graph neural networks. *arXiv preprint arXiv:2302.03266* (2023).
[62] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. 2019. DAG-GNN: DAG structure learning with graph neural networks. In *International Conference on Machine Learning*. PMLR, 7154–7163.
[63] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. 2021. From canonical correlation analysis to self-supervised graph neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 76–89.
[64] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. 2019. Prone: Fast and scalable network representation learning.. In *IJCAI*, Vol. 19. 4278–4284.
[65] Muhan Zhang and Pan Li. 2021. Nested graph neural networks. *Advances in Neural Information Processing Systems* 34 (2021), 15734–15747.
[66] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. 2019. Hierarchical Graph Pooling with Structure Learning. *arXiv preprint arXiv:1911.05954* (2019).
[67] Kangfei Zhao, Jeffrey Xu Yu, Qiyan Li, Hao Zhang, and Yu Rong. 2023. Learned sketch for subgraph counting: a holistic approach. *The VLDB Journal* (2023), 1–26.
[68] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. 2020. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*. PMLR, 11458–11468.
[69] Alexander Zhou, Yue Wang, and Lei Chen. 2023. Butterfly counting and bitruss decomposition on uncertain bipartite graphs. *The VLDB Journal* (2023), 1–24.
[70] Yingli Zhou, Yixiang Fang, Chenhao Ma, Tianci Hou, and Xin Huang. 2024. Efficient Maximal Motif-Clique Enumeration over Large Heterogeneous Information Networks. *Proceedings of the VLDB Endowment* 17, 11 (2024), 2946–2959.

A Appendix

A.1 Training Procedure Analysis

Based on the above discussion, we present the subgraph extraction algorithm via edge-ego net in Alg. 1. The algorithm takes the backbone graph G, the minimum motif size $n_{\min}$, and hop number k as the input, and outputs the subgraph set S_{sub}. The subgraph extraction based on ego net is similar and omitted here. For each edge in backbone graph G_{bb}, we extract the k-hop edge-ego net by retrieving the k-hop neighbors (lines 2-3). Next, it filters out the subgraphs whose size is smaller than that of the motif because the motif will never be isomorphic to a smaller graph (line 4). The remaining subgraphs will be added to S_{sub} (lines 6-7). Finally, the subgraph set is returned (line 8).

Algorithm 1: Edge-Ego Net Extraction

Input : Graph $G = (V, E)$, hop number k
Output : Subgraph Set S_{sub}

1 Initialize $S_{sub} \leftarrow \emptyset$
2 **for** $e(i, j)$ *in* E **do**
3 $\quad V_{ego}[i, j] \leftarrow \texttt{neighbor}(i, k) \cup \texttt{neighbor}(j, k)$
4 $\quad G_{ego}[i, j] \leftarrow \texttt{inducedSubgraph}(G, V_{ego}[i, j])$
5 $\quad$ Append $G_{ego}[i, j]$ to S_{sub}
6 **return** S_{sub}

Alg. 2 illustrates the structure learning process. ratio2rank$(\cdot)$ and thre2rank$(\cdot)$ denote the ratio-based edge selection strategy and threshold-based edge selection strategy, respectively.

Algorithm 2: Uncertain Subgraph Structure Learning

Input : Ego net $G_{ego} = (V_{ego}, E_{ego})$, selection strategy t, score threshold γ or ratio limit ρ
Output : Vertex feature matrix $\mathbf{X}$, edge set E_{ego}

1 Initialize $\mathbf{X} \leftarrow \mathbf{0} \in \mathbb{R}^{|V_{ego}| \times d}$
2 **foreach** $(i, j) \in E_{ego}$ **do**
$\mathbf{Z}(i, j) \leftarrow MLP([freq(i, j) || P(i, j)])$
3 $\mathbf{Z} \leftarrow \texttt{softmax}(\mathbf{Z})$
4 **if** t = ratio **then** index $\leftarrow \texttt{ratio2rank}(\mathbf{Z}, \rho)$
5 **else if** t = threshold **then** index $\leftarrow \texttt{thre2rank}(\mathbf{Z}, \gamma)$
6 $E_{ego} \leftarrow E_{ego}[\text{index}]$ // get top edges
7 $\mathbf{X} \leftarrow \texttt{filter}(\mathbf{X}, E_{ego})$
8 **return** $\mathbf{X}, E_{ego}$

The training procedure for one training epoch is illustrated in Alg. 3. In each epoch, the training set is separated into batches with size n_b. Before training, we employ edge-ego net extraction as a preprocessing step to obtain the initial subgraph set S_{sub} for the uncertain graph $\mathcal{G}$. We then perform uncertain subgraph structure learning on each subgraph to involve the uncertainty of edges and generate the representative possible worlds for the counting task (Line 4). With the obtained possible worlds, the target graph network can learn the representation $\mathbf{h}_{\mathcal{G}}$ of input uncertain graph $\mathcal{G}$ (Lines 5-9). The motif representation $\mathbf{h}_{\mathcal{M}}$ will be learned by motif network (Lines 11-15). With the obtained graph representation and motif representation, the predicted mean and variance value can be calculated by a multi-layer perceptron (Line 16). The loss is calculated by (17). The model parameter Θ will be updated by the sum of the loss in one batch.

Algorithm 3: The training procedure

Input : Training motif set $\mathcal{M}_t$, subgraph set $\mathcal{S}_{sub}$, component GNN layer number K_c, motif neural network layer number K_m , batch size n_b, trade-off parameter α
Output : Updated parameters Θ

1 Initialize global optimizer opt_Θ
2 Separate training set into batches $\{\mathcal{S}_b = \{\mathcal{M}^i\}\}$ of size n_b
3 **foreach** $G_{ego} \in \mathcal{S}_{sub}$ **do**
4 $\quad$ Update G_{ego} by Alg. 2 // USSL
5 $\quad$ **foreach** *layer* $l \leftarrow 1$ *to* K_c **do**
6 $\quad\quad$ **foreach** vertex $i \in V_{ego}$ **do** $\mathbf{h}_i^{(l)} \leftarrow$ Eq. (4)
7 $\quad\quad$ $\mathbf{h}_{G_{ego}}^{(l)} \leftarrow \texttt{Readout}(\mathbf{h}_i \mid i \in V_{ego})$
8 $\quad$ $\mathbf{h}_{G_{ego}} \leftarrow [\mathbf{h}_{G_{ego}}^{(1)} || \ldots || \mathbf{h}_{G_{ego}}^{(K_c)}]$
9 $\mathbf{h}_{\mathcal{G}} \leftarrow \texttt{Readout}(\mathbf{h}_{G_{ego}} \mid G_{ego} \in \mathcal{S}_{sub})$
10 **foreach** $\mathcal{S}_b \in \mathcal{M}_t$ **do**
11 $\quad$ **foreach** $\mathcal{M} \in \mathcal{S}_b$ **do**
12 $\quad\quad$ **foreach** *layer* $l \leftarrow 1$ *to* K_m **do**
13 $\quad\quad\quad$ **foreach** vertex $i \in \mathcal{M}$ **do** $\mathbf{h}_i^{(l)} \leftarrow$ Eq. (11)
14 $\quad\quad\quad$ $\mathbf{h}_{\mathcal{M}}^{(l)} \leftarrow \texttt{Sum}(\mathbf{h}_i^{(l)} \mid i \in V_{\mathcal{M}})$
15 $\quad\quad$ $\mathbf{h}_{\mathcal{M}} \leftarrow [\mathbf{h}_{\mathcal{M}}^{(1)} || \ldots || \mathbf{h}_{\mathcal{M}}^{(K_m)}]$
16 $\quad\quad$ $\hat{m}, \hat{v} \leftarrow MLP([\mathbf{h}_{\mathcal{M}} || \mathbf{h}_{\mathcal{G}}])$
17 $\quad\quad$ $\mathcal{L}_\Theta(\mathcal{M}) \leftarrow$ Eq. (17)
18 $\quad$ Update model parameters Θ by opt_Θ with $\sum_{\mathcal{M} \in S_b} \mathcal{L}_\Theta(\mathcal{M})$
19 **return** Θ

A.2 Time Complexity Analysis

The inference time complexity of each module is outlined as follows:

(1) Uncertain subgraph structure learning: $O(|E| \cdot |E_{ego}|)$, where $|E_{ego}|$ represents the maximum number of edges within one subgraph;
(2) Target graph neural network: $O(|E| \times |V_{ego}| \times d_{ego})$, with $|V_{ego}|$ and d_{ego} indicating the maximum number of vertices and the maximum degree in each subgraph, respectively;
(3) Motif neural network: $O(|V_{\mathcal{M}}| \times d_{\mathcal{M}})$, where $d_{\mathcal{M}}$ signifies the maximum degree of motif $\mathcal{M}$;
(4) Counting unit: $O(1)$.

k-hop ego net extraction, whose time complexity is $O(|E| \times |d_{max}|^k)$, is a pre-processing procedure and is not processed during inference. Therefore, the comprehensive inference time complexity of UnG-MoCha becomes $O(|E| \times (|E_{ego}| + |V_{ego}| \times d_{ego}) + |V_{\mathcal{M}}| \times d_{\mathcal{M}})$.

Hyper-parameters (e.g., hidden dimensions, layers) are excluded as they are adjustable constants unrelated to graph properties, though they affect execution time. With motif size fixed, as the edge number $|E|$ increases, UnG-MoCha exhibits approximately linear growth in inference time, demonstrating strong scalability.